列表控件和适配器

万永权

主要内容

Contents

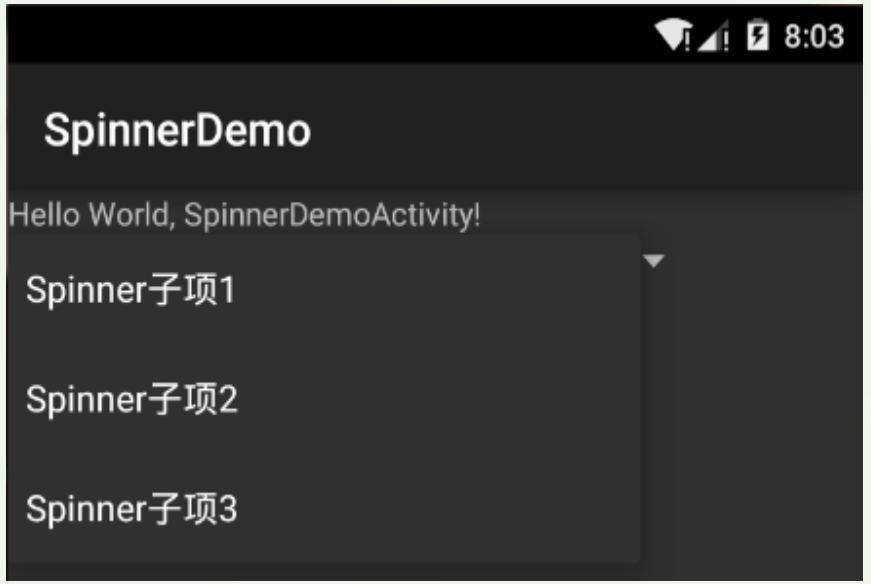
1.Spinner控件的使用

2.ListView控件的使用

5.2 界面控件

•5.2.4 Spinner

- 一种能够从多个选项中选一选项的控件，使用浮动菜单为用户提供选择
- 类似于桌面程序的组合框（ComboBox）



5.2 界面控件

•5.2.4 Spinner

- SpinnerDemo在XML文件中的代码

```
1.  <TextView  android:id="@+id/TextView01"
2.     android:layout_width=“match_parent"
3.     android:layout_height="wrap_content"
4.     android:text="@string/hello"/>
5.  <Spinner android:id="@+id/Spinner01"
6.  android:layout_width="300dip"
7.  android:layout_height="wrap_content">
8.  </Spinner>
```

- 第5行使用<Spinner>标签声明了一个Spinner控件
- 第6行代码中指定了该控件的宽度为300dip
- dip是设备独立像素，不同设备有不同 的现实效果

5.2 界面控件

- 在SpinnerDemo.java文件中，定义一个ArrayAdapter适配器，在ArrayAdapter中添加需要在Spinner中可以选择的内容
- 适配器绑定界面控件和底层数据，若底层数据更改了，用户界面也相应修改显示内容，就不需要应用程序再监视，从而极大的简化的代码的复杂性

```
1. Spinner spinner = (Spinner) findViewById(R.id.Spinner01);
2. List<String> list  = new ArrayList<String>();
3. list .add("Spinner子项1");
4. list .add("Spinner子项2");
5. list .add("Spinner子项3");
6. ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
   android.R.layout.simple_spinner_item, list );
7. adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
8. spinner.setAdapter(adapter);
```

5.2 界面控件

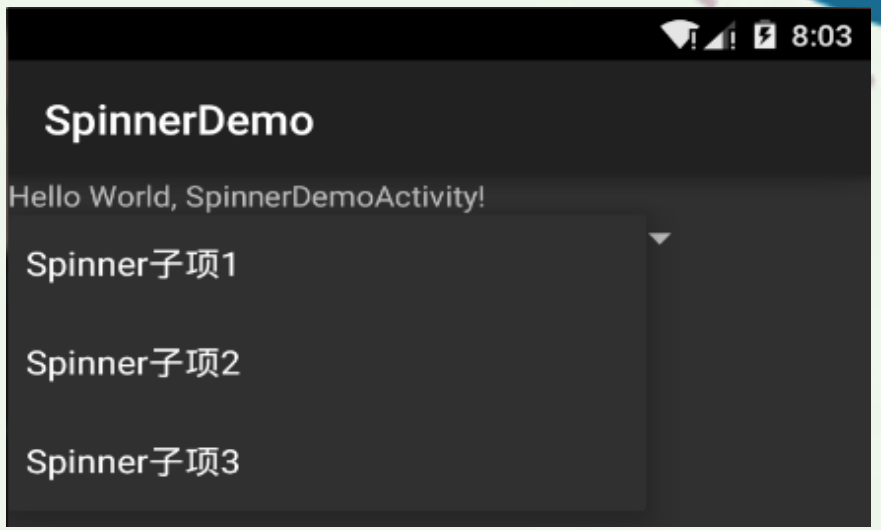
•5.2.4 Spinner

- 第2行代码建立了一个数组列表（ArrayList），这种数组列表可以根据需要进行增减
- <String>表示数组列表中保存的是字符串类型的数据
- 在代码的第3、4、5行中，使用add()函数分别向数组列表中添加3个字符串
- 第6行代码建立了一个ArrayAdapter的数组适配器，数组适配器能够将界面控件和底层数据绑定在一起
- 第8行代码实现绑定过程，所有ArrayList中的数据，将显示在Spinner的浮动菜单中
- 第7行代码设定了Spinner的浮动菜单的显示方式，其中，android.R.layout.simple_spinner_dropdown_item是Android系统内置的一种浮动菜单

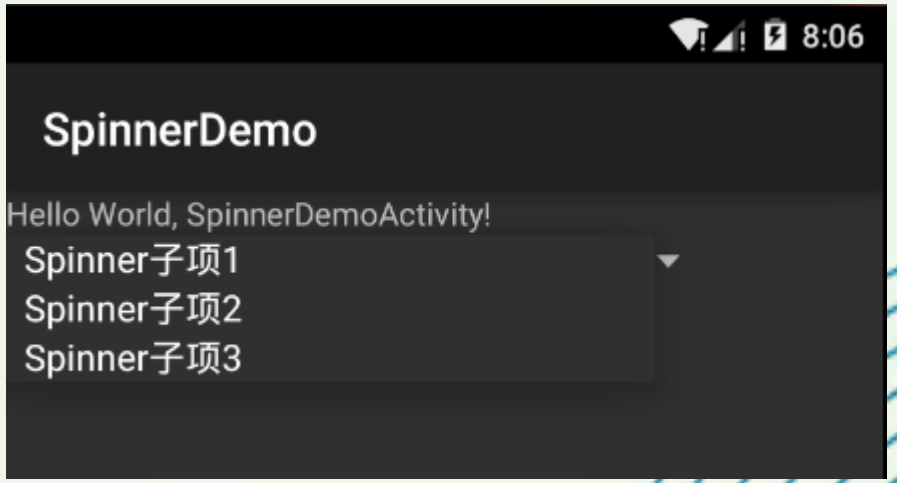
5.2 界面控件

•5.2.4 Spinner

- Spinner的浮动菜单的显示方式
  - android.R.layout.simple_spinner_dropdown_item



  - android.R.layout.simple_spinner_item



5.2 界面控件

•5.2.5 ListView

- ListView是一种用于垂直显示的列表控件，如果显示内容过多，则会出现垂直滚动条
- ListView能够通过适配器将数据和自身绑定，在有限的屏幕上提供大量内容供用户选择,所以是经常使用的用户界面控件
- ListView支持点击事件处理，用户可以用少量的代码实现复杂的选择功能

ListView属性

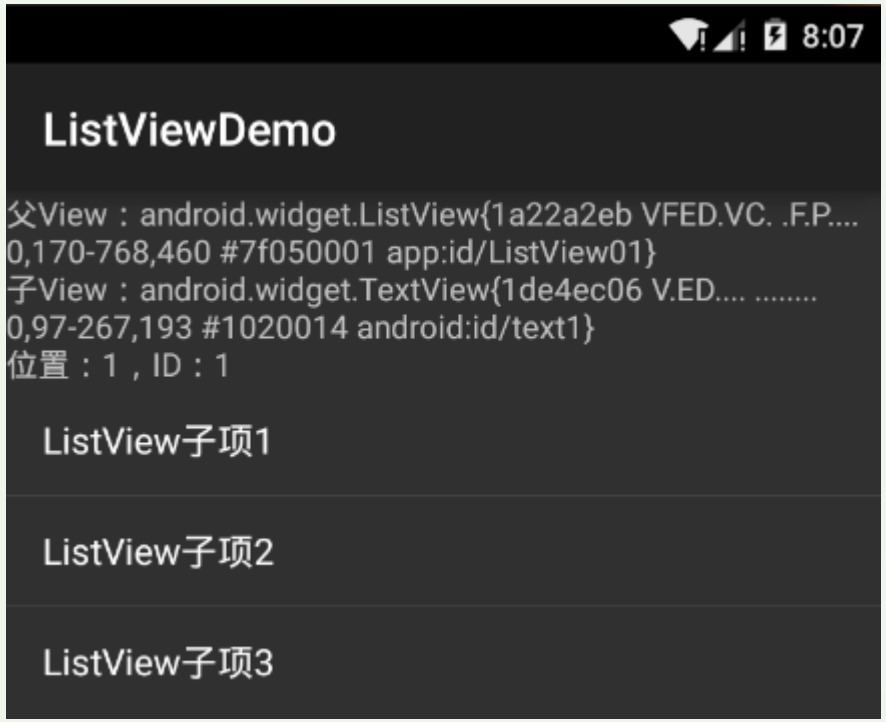
ListView支持的XML属性

XML属性	描述
android:divider	用于为列表视图设置分隔条，既可以用颜色分隔，也可以用Drawable资源分隔
android:dividerHeight	用于设置分隔条的高度
android:entries	用于通过数组资源为ListView指定列表项
android:footerDividersEnabled	用于设置是否在footer View之前绘制分隔条，默认值为true，设置为false时，表示不绘制。使用该属性时，需要通过ListView组件提供的addFooterView()方法为ListView设置footer View
android:headerDividersEnabled	用于设置是否在header View之后绘制分隔条，默认值为true，设置为false时，表示不绘制。使用该属性时，需要通过ListView组件提供的addHeaderView()方法为ListView设置header View

5.2 界面控件

•5.2.5 ListView

- 建立一个“ListViewDemo”程序，包含四个控件，从上至下分别为TextView01、ListView01、 ListView02和 ListView03



5.2 界面控件

•5.2.5 ListView

- ListViewDemo在XML文件中的代码

```
1.  <TextView  android:id="@+id/TextView01"
2.  android:layout_width=“match_parent"
3.  android:layout_height="wrap_content"
4.  android:text="@string/hello" />
5.  <ListView android:id="@+id/ListView01"
6.  android:layout_width="wrap_content"
7.  android:layout_height="wrap_content">
8.  </ListView>
```

5.2 界面控件

•5.2.5 ListView

- 在ListViewDemo.java文件中，首先需要为ListView创建适配器，并添加ListView中所显示的内容

```
1.  final TextView textView = (TextView)findViewById(R.id.TextView01);
2.  ListView listView = (ListView)findViewById(R.id.ListView01);
3.  List<String> list  = new ArrayList<String>();
4.  list.add("ListView子项1");
5.  list.add("ListView子项2");
6.  list.add("ListView子项3");
7.  ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, list );
8.  listView.setAdapter(adapter);
```

- 第2行代码通过ID引用了XML文件中声明的ListView
- 第7行代码声明了适配器ArrayAdapter，第三个参数list说明适配器的数据源为数组列表
- 第8行代码将ListView和适配器绑定

5.2 界面控件

•5.2.5 ListView

- 下面的代码声明了ListView子项的点击事件监听器，用以确定用户在ListView中，选择的是哪一个子项

```
1. AdapterView.OnItemClickListener listViewListener = new
   AdapterView.OnItemClickListener(){
2. @Override
3. public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
4. String msg = "父View： "+arg0.toString()+"\n"+"子View： "+arg1.toString()+"\n"+"位置：
   "+String.valueOf(arg2)+"， ID： "+String.valueOf(arg3);
5. textView.setText(msg);
6. }};
7. listView.setOnItemClickListener(listViewListener);
```

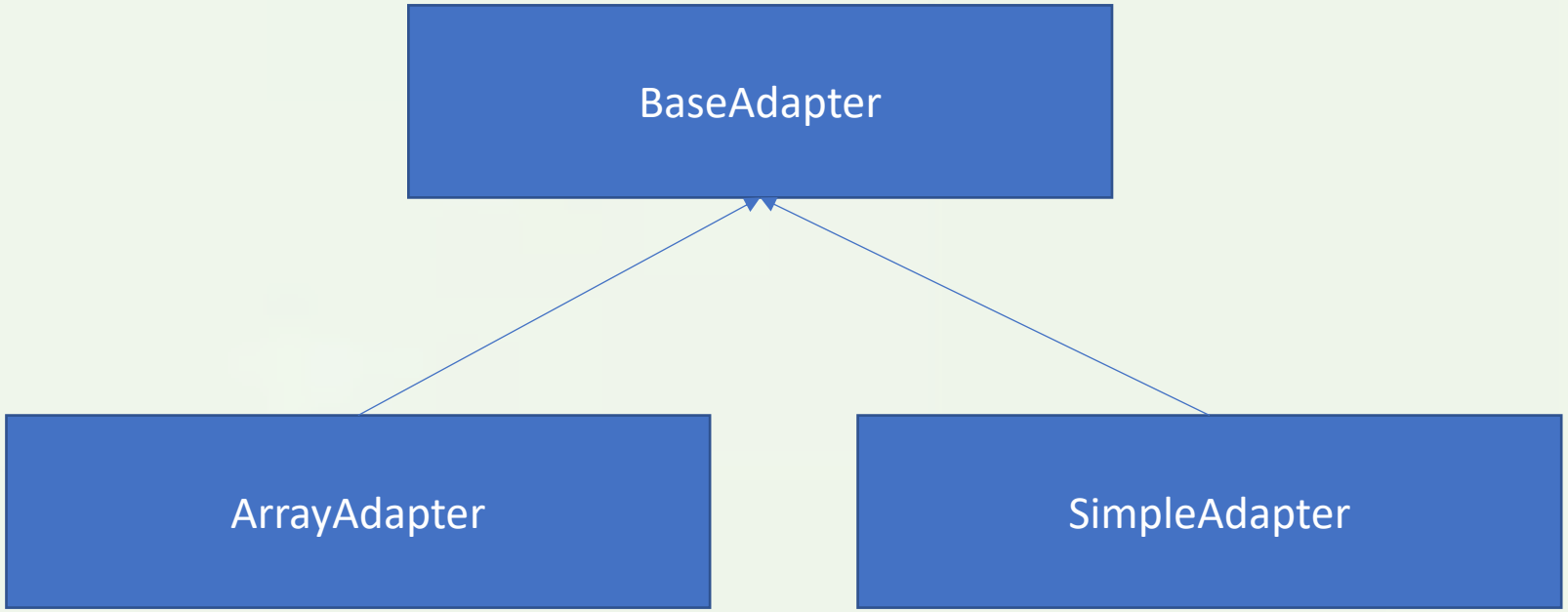
5.2 界面控件

•5.2.5 ListView

- 第1行的AdapterView.OnItemClickListener是ListView子项的点击事件监听器，同样是一个接口，需要实现onItemClick()函数。在ListView子项被选择后，onItemClick()函数将被调用
- 第3行的onItemClick()函数中一共有四个参数，参数0表示适配器控件，就是ListView；参数1表示适配器内部的控件，是ListView中的子项；参数2表示适配器内部的控件，也就是子项的位置；参数3表示子项的行号
- 第4行和第5行代码用于显示信息，选择子项确定后，在TextView中显示子项父控件的信息、子控件信息、位置信息和ID信息
- 第7行代码是ListView指定刚刚声明的监听器

适配器

Adapter



常用数据适配器（Adapter）

常用数据适配器

数据适配器是数据与视图之间的桥梁，它类似于一个转换器，将复杂的数据转换成用户可以接受的方式进行呈现。

常用的数据适配器：

- BaseAdapter：基本的适配器
- SimpleAdapter：继承自BaseAdapter
- ArrayAdapter：也是BaseAdapter的子类

1. BaseAdapter

BaseAdapter顾名思义是基本的适配器。它实际上是一个抽象类，通常在自定义适配器时会继承BaseAdapter，该类拥有四个抽象方法，根据这几个抽象方法对ListView控件进行数据适配。BaseAdapter中的4个抽象方法如下表所示。

方法名称	功能描述
public int getCount()	获取列表条目的总数
public Object getItem(int position)	根据position（位置）获取某个条目的对象
public long getItemId(int position)	根据position（位置）获取某个条目的id
public View getView(int position, View convertView, ViewGroup parent)	获取相应position对应的条目视图，position是当前条目的位置，convertView用于复用旧视图，parent用于加载XML布局。

SimpleAdapter

常用数据适配器（Adapter）

2. SimpleAdapter

SimpleAdapter继承BaseAdapter，实现了BaseAdapter的四个抽象方法并进行封装。

SimpleAdapter的构造方法的具体信息如下：

```
public SimpleAdapter(Context context, List<? extends Map<String, ?>> data,int resource,
String[] from, int[] to)
```

在SimpleAdapter()构造方法中的5个参数的含义如下：

- context：表示上下文对象。
- data：数据集合，data中的每一项对应ListView控件中的条目的数据。
- resource：条目布局的资源id。
- from：Map集合中的key值。
- to：条目布局中对应的控件。

使用SimpleAdapter

- 1.设计列表项（List Item）的布局；
- 2. 准备数据；
- 3. 创建SimpleAdapter；
- 4. 在ListView上设置Adapter；

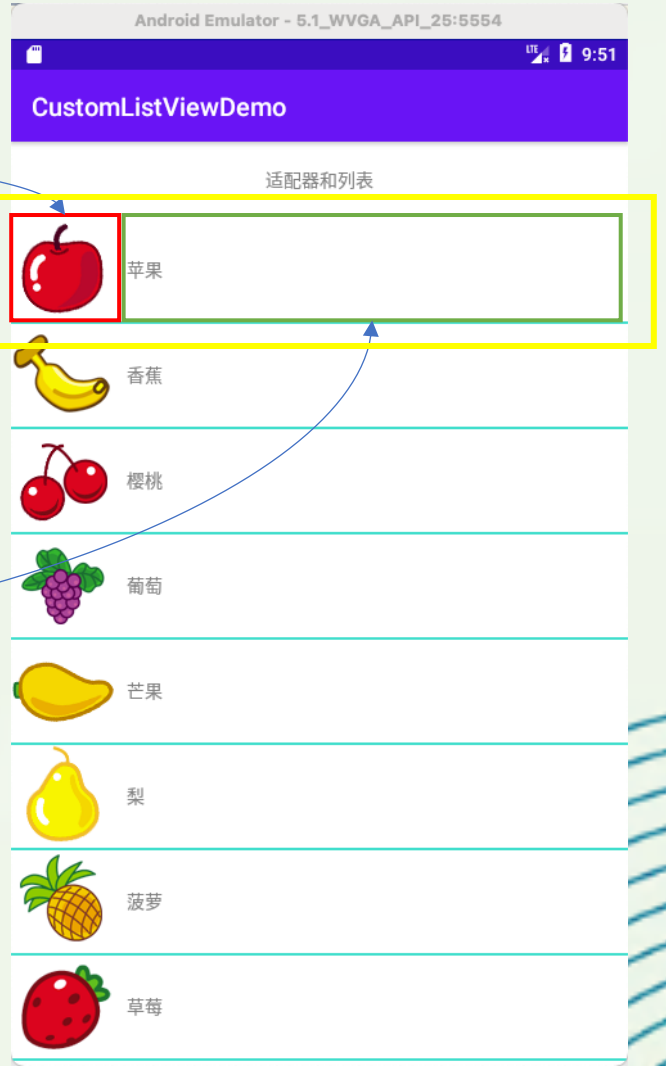
ListView Item 的布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/fruit_image"
        android:layout_width="80dp"
        android:layout_height="80dp" />

    <TextView
        android:id="@+id/fruit_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="10dp" />

</LinearLayout>
```



常用数据适配器（Adapter）

3. ArrayAdapter

ArrayAdapter也是BaseAdapter的子类，用法与SimpleAdapter类似，开发者只需要在构造方法里面传入相应参数即可。ArrayAdapter通常用于适配TextView控件，ArrayAdapter有多个构造方法，构造方法的具体信息如下所示。

```
public ArrayAdapter(Context context,int resource) ;
public ArrayAdapter(Context context,int resource, int textViewResourceId) ;
public ArrayAdapter(Context context,int resource,T[] objects) ;
public ArrayAdapter(Context context,int resource,int textViewResourceId,T[] objects);
public ArrayAdapter(Context context,int resource,List<T> objects) ;
public ArrayAdapter(Context context,int resource,int textViewResourceId, List<T> objects)
```

自定义控件

自定义控件的步骤

- 1.设计布局
- 2.定义数据类
- 3.定义控件的适配器
- 4.在其他布局中使用该控件

自定义ListView (使用 ArrayAdapter)

- 定制ListView界面
- 1：新建一个数据类；　（这里是Fruit类）
- 2：新建一个列表item布局（fruit_item.xml）
- 3：　自定义一个适配器（FruitAdapter）
- 4：　在合适的布局中使用自定义的ListView

01

新建一个数据类

ListView

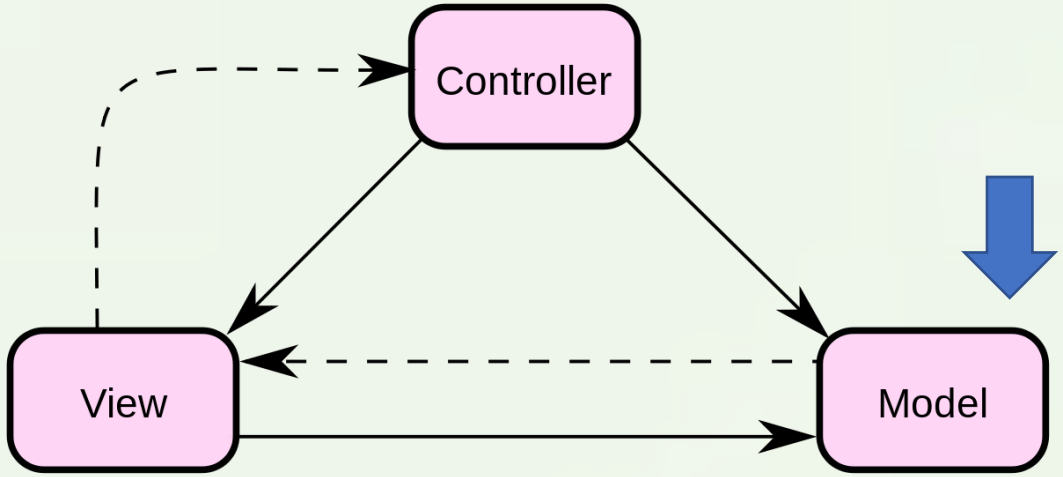
1.Fruit类

定义水果类，里面包含名字和图片

```
public class Fruit {
    private String name;
    private int imageId;
    private double price;
    public Fruit(String name ,int imageId, double price){
        this.name = name ;
        this.imageId = imageId;
        this.price = price;
    }
    public String getName(){

        return name;
    }
    public int getImageId(){
        return imageId;
    }

    public double getPrice() {
        return price;
    }
}
```



02

新建一个列表item布局

ListView

2. Fruit_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/fruit_image"
        android:layout_width="80dp"
        android:layout_height="80dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="10dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/fruit_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="5dp" />

        <TextView
            android:id="@+id/fruit_price"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp" />
    </LinearLayout>

</LinearLayout>
```

03

自定义一个适配器

继承BaseAdapter

- 继承BasedAdapter并重写它的四个抽象方法：
  - public int getCount()：得到Item的总数。
  - public Object getItem(int position)：根据position得到某个Item的对象。
  - public long getItemId(int position)：根据position得到某个Item的id
  - public View getView(int position, View convertView, ViewGroup parent)：得到相应position对应的Item视图，position当前Item的位置，convertView复用的View对象。

ArrayAdapter

- 继承ArrayAdapter并重写它的1个抽象方法：
  - public View getView(int position, View convertView, ViewGroup parent)：
    - 得到相应position对应的Item视图；
    - position：当前Item的位置；
    - convertView复用的View对象。

ListView

3. 定义适配器FruitAdapter

```
public class FruitAdapter extends ArrayAdapter<Fruit> {
    private int resourceId;
    public FruitAdapter(Context context, int textViewResourceId, List<Fruit>objects){
        super(context,textViewResourceId,objects);
        resourceId = textViewResourceId;
    }
public View getView(int position,View convertView,ViewGroup parent){
        Fruit fruit = getItem(position);
 View view = LayoutInflater.from(getContext()).inflate(resourceId,parent,false);
        ImageView fruitImage = (ImageView) view.findViewById(R.id.fruit_image);
       TextView fruitName = (TextView) view.findViewById(R.id.fruit_name);
        fruitName.setText(fruit.getName());
       fruitImage.setImageResource((fruit.getImageId()));
      return view;
}
```

04

在合适的布局中使用自定义的ListView

LIstView

4.主函数调用(1)

```
public class MainActivity extends AppCompatActivity {
    private List<Fruit> fruitList = new ArrayList<>();
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initFruits();
        FruitAdapter adapter = new FruitAdapter(MainActivity.this,
R.layout.fruit_item, fruitList);
        ListView listView = (ListView) findViewById(R.id.list_View);
        listView.setAdapter(adapter);
```

LIstView

4. 主函数调用(2)

```
private void initFruits(){
        for(int i=0;i<2;i++){
            Fruit apple = new Fruit("apple",R.drawable.apple_pic);
            fruitList.add(apple);
            Fruit banana = new
Fruit("Banana",R.drawable.banana_pic);
            fruitList.add(banana);
            Fruit orange = new
Fruit("Orange",R.drawable.orange_pic);
            fruitList.add(orange);
            Fruit watermelon = new
Fruit("Watermelon",R.drawable.watermelon_pic);
            fruitList.add(watermelon);
            Fruit pear = new Fruit("Pear",R.drawable.pear_pic);
            fruitList.add(pear);
            Fruit grape = new Fruit("Grape",R.drawable.grape_pic);
            fruitList.add(grape);
            Fruit pineapple = new
Fruit("Pineapple",R.drawable.pineapple_pic);
            fruitList.add(pineapple);
            Fruit strawberry = new
Fruit("Strawberry",R.drawable.strawberry_pic);
            fruitList.add(strawberry);
            Fruit cherry = new
Fruit("Cherry",R.drawable.cherry_pic);
            fruitList.add(cherry);
            Fruit mango = new Fruit("Mango",R.drawable.mango_pic);
            fruitList.add(mango);
        }
    }
```

练习

- 完成如下效果，文字和图片可自选，界面中请显示学号姓名，界面上交运行截图即可，图片文件以学号+姓名+作业

实战演练—购物商城

接下来我们通过一个购物商城的案例来演示如何通过ListView控件与数据适配器显示一个商品信息的列表。本案例的界面效果如下图所示。

1. **创建程序：**
   ①创建一个名为ListView的应用程序
   ②指定包名为cn.itcast.listview
2. **导入界面图片：**将界面需要的图片导入到drawable-hdpi文件夹中
3. **放置界面控件：**
   ①放置1个TextView控件
   ②放置1个ListView控件
4. **创建列表条目：**创建列表条目界面的布局文件list_item.xml
5. **实现界面显示效果：**
   ①在MainActivity中创建适配器MyBaseAdapter
   ②实现对ListView控件的数据适配

实战演练—购物商城

优化ListView加载数据逻辑

运行前面的购物商城程序后，当ListView控件上加载的条目过多，并快速滑动该列表控件时，界面会出现卡顿的现象，出现这个现象的原因如下：

（1）当滑动屏幕时，不断地创建条目对象

（2）不断执行findViewById()方法初始化界面控件

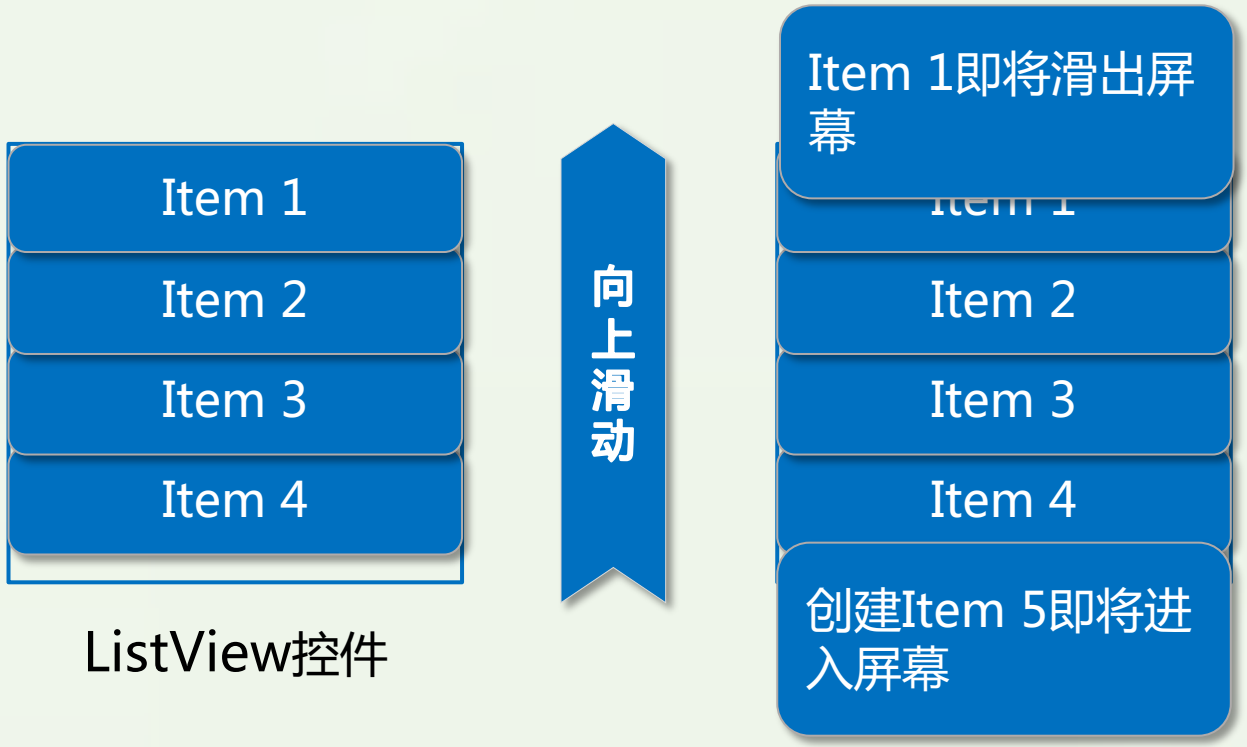
由于上述两点原因，我们需要对ListView控件进行优化，优化目的是使ListView控件在快速滑动时不再重复创建条目对象，减少内存的消耗和屏幕渲染的处理。优化方式有以下两种：

（1）使用ViewHolder类

（2）复用convertView

优化ListView控件加载数据

为了防止数据量过大造成内存溢出，在使用ListView时通常会进行优化，优化方式中的其中一种是复用convertView。



优化ListView控件加载数据